Advances in Intelligent Systems and Computing 238

Jeng-Shyang Pan
Pavel Krömer
Václav Snášel *Editors*

# Genetic and Evolutionary Computing

Proceedings of the Seventh International Conference on Genetic and Evolutionary Computing, ICGEC 2013, August 25–27, 2013 - Prague, Czech Republic

Springer

Advances in Intelligent Systems and Computing

Volume 238

*Series Editor*

Janusz Kacprzyk, Warsaw, Poland

For further volumes:
http://www.springer.com/series/11156

Jeng-Shyang Pan · Pavel Krömer
Václav Snášel
Editors

# Genetic and Evolutionary Computing

Proceedings of the Seventh International Conference on Genetic and Evolutionary Computing, ICGEC 2013, August 25–27, 2013 - Prague, Czech Republic

Springer

*Editors*
Jeng-Shyang Pan
Department of Electronic Engineering
National Kaohiung University of Applied Sciences
Kaohsiung
Taiwan R.O.C.

Václav Snášel
Department of Computer Science
Faculty of Ele. Eng. and Computer Science
VŠB-TUO
Ostrava-Poruba
Czech Republic

Pavel Krömer
Department of Computer Science
Faculty of Ele. Eng. and Computer Science
VŠB-TUO
Ostrava-Poruba
Czech Republic

ISSN 2194-5357 ISSN 2194-5365 (electronic)
ISBN 978-3-319-01795-2 ISBN 978-3-319-01796-9 (eBook)
DOI 10.1007/978-3-319-01796-9
Springer Cham Heidelberg New York Dordrecht London

Library of Congress Control Number: 2013945191



Printed on acid-free paper

Springer is part of Springer Science+Business Media (www.springer.com)

# Preface

This volume of Advances in Intelligent Systems and Computing contains accepted papers presented at ICGEC 2013, the 7th International Conference on Genetic and Evolutionary Computing. The conference this year was technically co-sponsored by The Waseda University in Japan, Kaohsiung University of Applied Science in Taiwan, and VŠB-Technical University of Ostrava. ICGEC 2013 was held in Prague, Czech Republic. Prague is one of the most beautiful cities in the world whose magical atmosphere has been shaped over ten centuries. Places of the greatest tourist interest are on the Royal Route running from the Powder Tower through Celetná Street to Old Town Square, then across Charles Bridge through the Lesser Town up to the Hradčany Castle. One should not miss the Jewish Town, and the National Gallery with its fine collection of Czech Gothic art, collection of old European art, and a beautiful collection of French art.

The conference was intended as an international forum for the researchers and professionals in all areas of genetic and evolutionary computing. The main topics of ICGEC 2013 included Intelligent Computing, Evolutionary Computing, Genetic Computing, and Grid Computing.

The organization of the ICGEC 2013 was entirely voluntary. The review process required an enormous effort from the members of the International Technical Program Committee, and we would therefore like to thank all its members for their contribution to the success of this conference. We would like to express our sincere thanks to the invited session organizers, to the host of ICGEC 2013, VŠB – Technical University of Ostrava, and to the publisher, Springer, for their hard work and support in organizing the conference.

August 2013

Jeng-Shyang Pan
Pavel Krömer
Václav Snášel

# Organization

## Conference Chair

| | |
|---|---|
| Václav Snášel | VŠB - Technical University of Ostrava |

## Program Chair

| | |
|---|---|
| Yong Xu | Harbin Institute of Technology Shenzhen Graduate School, China |

## Advisory Chair

| | |
|---|---|
| Jeng-Shyang Pan | Kaohsiung University of Applied Sciences, Taiwan |

## Invited Session Chair

| | |
|---|---|
| Wu-Chih Hu | National Penghu University of Science and Technology, Taiwan |

## Honorary Chairs

| | |
|---|---|
| Junzo Watada | Waseda University, Japan |
| Bin-Yih Liao | Kaohsiung University of Applied Sciences, Taiwan |

## Publicity Chair

| | |
|---|---|
| Jaroslav Pokorný | Charles University, Czech Republic |

## International Program Committee

| | |
|---|---|
| Akira Asano | Hiroshima University, Japan |
| Mohsen Askari | University of Technology Sydney, Australia |
| Dariusz Barbucha | Gdynia Maritime University, Poland |
| Jonathan Hoyin Chan | King Mongkut's University of Technology Thonburi, Thailand |
| Chin-Chen Chang | Feng Chia University, Taiwan |
| Feng-Cheng Chang | Tamkang University, Taiwan |
| Yueh-Hong Chen | Far East University, Taiwan |
| Chao-Chun Chen | National Cheng Kung University, Taiwan |
| Rung-Ching Chen | Chaoyang University of Technology, Taiwan |
| Shi-Jay Chen | National United University, Taiwan |
| Pei-Yin Chen | National Cheng Kung University, Taiwan |
| Tsung-Che Chiang | National Taiwan Normal University, Taiwan |
| Shu-Chuan Chu | Cheng-Shiu University, Taiwan |
| Yi-Nung Chung | National Changhua University of Education, Taiwan |
| Maurice Clerc | Independent Consultant, France |
| Martine De Cock | Ghent University, Belgium |
| Jose Alfredo F. Costa | Federal University, Brazil |
| Zhihua Cui | Taiyuan University of Science and Technology, China |
| Alexander A. Frolov | Russian Academy of Sciences, Russia |
| Amparo Fúster-Sabater | Spanish Scientific Research Council, Spain |
| Petr Gajdoš | VŠB-Technical University of Ostrava, Czech Republic |
| Xiao-Zhi Gao | Helsinki University of Technology, Finland |
| Alexander Gelbukh | National Polytechnic Institute, Mexico |
| Gheorghita Ghinea | Brunel University, UK |
| Massimo De Gregorio | Istituto di Cibernetica, Italy |
| Stefanos Gritzalis | University of the Aegean, Greece |
| Ramin Halavati | Sharif University of Technology, Iran |
| Enrique Herrera-Viedma | University of Granada, Spain |
| Jiun-Huei Ho | Cheng Shiu University, Taiwan |
| Chian C. Ho | National Yunlin University of Science & Technology, Taiwan |
| Wei-Chiang Hong | Oriental Institute of Technology, Taiwan |
| Cheng-Hsiung Hsieh | Chaoyang University of Technology, Taiwan |
| Chien-Chang Hsu | Fu Jen Catholic University, Taiwan |
| Shu-Hua Hu | Jinwen University of Science and Technology, Taiwan |
| Yongjian Hu | South China University of Technology, China |

| | |
|---|---|
| Ming-Wen Hu | Tamkang University, Taiwan |
| Wu-Chih Hu | National Penghu University of Science and Technology, Taiwan |
| Deng-Yuan Huang | Dayeh University, Taiwan |
| Tien-Tsai Huang | Lunghwa University of Science and Technology, Taiwan |
| Hsiang-Cheh Huang | National Kaohsiung University, Taiwan |
| Yung-Fa Huang | Chaoyang University of Technology, Taiwan |
| Dusan Husek | Academy of Sciences of the Czech Republic, Czech Republic |
| Donato Impedovo | University of Bari, Italy |
| Albert B. Jeng | Jinwen University of Science and Technology, Taiwan |
| Isabel S. Jesus | Institute of Engineering of Porto, Portugal |
| Jie Jing | Zhejiang University of Technology, China |
| Estevam R. Hruschka Jr. | Federal University of Sao Carlos, Brazil |
| Muhammad Khurram Khan | King Saud University, Kingdom of Saudi Arabia |
| Mario Koeppen | Kyushu Institute of Technology, Japan |
| Gabriella Kokai | Fraunhofer Institute for Integrated Circuits, Germany |
| Miloš Kudělka | VŠB-Technical University of Ostrava, Czech Republic |
| Hsu-Yang Kung | National Pingtung University of Science and Technology, Taiwan |
| Yau-Hwang Kuo | National Cheng Kung University, Taiwan |
| Kun-Huang Kuo | Chienkuo Technology University, Taiwan |
| Weng Kin Lai | MIMOS Berhad, Malaysia |
| Jenn-Kaie Lain | National Yunlin University of Science and Technology, Taiwan |
| Kwok-Yan Lam | Tsinghua University, China |
| Jouni Lampinen | University of Vaasa, Finland |
| Sheau-Dong Lang | Central Florida University, USA |
| Chang-Shing Lee | National University of Tainan, Taiwan |
| Jung-San Lee | Feng Chia University, Taiwan |
| Huey-Ming Lee | Chinese Culture University, Taiwan |
| Shie-Jue Lee | National Sun Yat-Sen University, Taiwan |
| Jorge E. Núñez Mc Leod | Universidad Nacional de Cuyo, Argentina |
| Yue Li | Nankai University, China |
| Chang-Tsun Li | University of Warwick, UK |
| Jun-Bao Li | Harbin Institute of Technology, China |
| Guan-Hsiung Liaw | I-Shou University, Taiwan |
| Tsung-Chih Lin | Feng-Chia University, Taiwan |
| Lily Lin | China University of Technology, Taiwan |
| Yuh-Chung Lin | Tajen University, Taiwan |

| | |
|---|---|
| Wen-Yang Lin | National University of Kaohsiung, Taiwan |
| Jim-Min Lin | Feng Chia University, Taiwan |
| Wei-Cheng Lin | National Kaohsiung University of Applied Sciences, Taiwan |
| Chia-Chen Lin | Providence University, Taiwan |
| Tianhua Liu | Shenyang Normal University, China |
| Yu-Lung Lo | Chaoyang University of Technology, Taiwan |
| Heitor Silverio Lopes | Federal University of Technology Parana, Brazil |
| Yuh-Yih Lu | Minghsin University of Science and Technology, Taiwan |
| Pavel Moravec | VŠB-Technical University of Ostrava, Czech Republic |
| Petr Musilek | University of Alberta, Canada |
| Marco Mussetta | Politecnico di Milano, Italy |
| Kazumi Nakamatsu | University of Hyogo, Japan |
| Jlio Cesar Nievola | Pontificia Universidade Catolica do Parana, Brazil |
| Yusuke Nojima | Osaka Prefecture University, Japan |
| Isabel L. Nunes | Universidade Nova Lisboa, Portugal |
| Eliska Ochodkova | VŠB-Technical University of Ostrava, Czech Republic |
| Jae C. Oh | Syracuse University, USA |
| S.N. Omkar | Indian Institute of Science, Indian |
| Djamila Ouelhadj | University of Portsmouth, UK |
| Mauricio Papa | University of Tulsa, USA |
| Marek Penhaker | VŠB-Technical University of Ostrava, Czech Republic |
| Sylvain Piechowiak | University of Valenciennes, France |
| Jan Platoš | VŠB-Technical University of Ostrava, Czech Republic |
| Aurora Trinidad Ramirez Pozo | Federal University of Parana, Brazil |
| Andri Riid | Tallinn University of Technology, Estonia |
| Selva S. Rivera | Universidad Nacional de Cuyo, Argentina |
| Lotfi Ben Romdhane | University of Monastir, Tunisia |
| G. Sainarayanan | ICT Academy of Tamil Nadu, India |
| Luciano Sanchez | Oviedo University, Spain |
| Yung-Jong Shiah | Kaohsiung Medical University, Taiwan |
| S.N. Singh | Indian Institute of Technology Kanpur, India |
| Georgios Ch. Sirakoulis | Democritus University of Thrace, Greece |
| Mariacarla Staffa | Universit degli Studi di Napoli Federico II, Italy |
| Jin-Shieh Su | Chinese Culture University, Taiwan |
| Shing Chiang Tan | Multimedia University, Malaysia |
| Hui-Chin Tang | National Kaohsiung University, Taiwan |
| Izzettin Temiz | Gazi University, Turkey |

| | |
|---|---|
| Chen-Da Tsai | Far East University, Taiwan |
| Tsung-Han Tsai | National Central University, Taiwan |
| Eiji Uchino | Yamaguchi University, Japan |
| Sebastin Ventura | University of Cordoba, Spain |
| Brijesh Verma | Central Queensland University, Australia |
| Michael N. Vrahatis | University of Patras, Greece |
| Shiuh-Jeng Wang | Central Police University, Taiwan |
| Feng Wang | Kunming University of Science and Technology, China |
| Ling Wang | Tsinghua University, China |
| Lidong Wang | National Computer Network Emergency Technical Team/ Coordination Center of China, China |
| Di Wang | Khalifa University, UAE |
| Yin Chai Wang | University Malaysia Sarawak, Malaysia |
| Kwok-Wo Wong | City University of Hong Kong, Hong Kong |
| Michal Wozniak | Wroclaw University of Technology, Poland |
| Ruqiang Yan | Southeast University, China |
| Chyuan-Huei Thomas Yang | Hsuan Chuang University, Taiwan |
| Chung-Huang Yang | National Kaohsiung Normal University, Taiwan |
| Sheng-Yuan Yang | St. John's University, Taiwan |
| Li Yao | University of Manchester, UK |
| Jui-Feng Yeh | National Chiayi University, Taiwan |
| Show-Jane Yen | Mining Chuan University, Taiwan |
| Fa-xin Yu | Zhejiang University, China |
| Ming Yuchi | Huazhong University of Science and Technology, China |
| Ivan Zelinka | VŠB-Technical University of Ostrava, Czech Republic |
| Zhigang Zeng | Wuhan University of Technology, China |
| Xiao-Jun Zeng | University of Manchester, UK |
| Zhiyong Zhang | Henan University of Science and Technology, China |
| Yong Zhang | Shenzhen University, China |
| Xinpeng Zhang | Shanghai University, China |
| Yongping Zhang | Hisilicon Technologies Co., Ltd, China |

## Sponsoring Institutions

VŠB - Technical University of Ostrava, Czech Republic
Waseda University, Japan
Kaohsiung University of Applied Science,Taiwan

# Contents

## Advances in Genetic and Evolutionary Computing

## Multimedia Innovative Computing

## Innovative Intelligent Internet Technology



## Multimedia Social Networks and Soft Computing

## Pattern Recognition and Multimedia Signal Processing



## High Speed Computation and Applications in Information Security Systems

# Forecast Models of Partial Differential Equations Using Polynomial Networks

Ladislav Zjavka

VŠB-Technical University of Ostrava, IT4innovations Ostrava, Czech Republic
ladislav.zjavka@vsb.cz

**Abstract.** Unknown data relations can describe lots of complex systems through partial differential equation solutions of a multi-parametric function approximation. Common neural network techniques of pattern classification or function approximation problems in general are based on whole-pattern similarity relationships of trained and tested data samples. They apply input variables of only absolute interval values, which may cause problems by far various training and testing data ranges. Differential polynomial neural network is a new type of neural network developed by the author, which constructs and substitutes an unknown general sum partial differential equation, defining a system model of dependent variables. It generates a total sum of fractional polynomial terms defining partial relative derivative dependent changes of some combinations of input variables. This type of regression is based only on trained generalized data relations. The character of relative data allows processing a wider range of test interval values than defined by the training set. The characteristics of differential equation solutions also in general facilitate a greater variety of model forms than allow standard soft computing methods.

**Keywords:** polynomial neural network, partial differential equation composition, sum relative derivative term, multi-parametric function approximation.

## 1 Introduction

Differential equation solutions allow define models for a variety of pattern recognition [10] and primarily function approximation problems, applying genetic programming (GP) techniques [3] or an artificial neural network (ANN) construction [9]. A common ANN operating principle is based on entire similarity relationships of new presented input patterns with the trained ones. It does not allow for eventual forthright data relations of variables, which might define a generalized model. Common soft-computing techniques utilize only absolute interval values of input variables, which are not able to describe a wider range of applied data [1]. The generalization from the training data set may be difficult or problematic if the model has not been trained with inputs around the range covered testing data [2]. If training data involve relations, which may become stronger or weaker character, the network

J.-S. Pan et al. (eds.), *Genetic and Evolutionary Computing*,
Advances in Intelligent Systems and Computing 238,
DOI: 10.1007/978-3-319-01796-9_1, 

model could generalize it into wide-range valid values. Differential polynomial neural network (D-PNN) is a new neural network type, which creates and resolves an unknown partial differential equation (DE) following a data description of a multi-parametric function approximation. A general DE is substituted producing sum of fractional polynomial derivative terms, forming a system model of dependent variables. In contrast with the ANN functionality, each neuron can direct take part in the network total output calculation, which is formed by the sum of active neuron outputs. The study tried to create a neural network, which function approximation is based on any dependent data relations. ANN solutions do not provide model specifications in the form of a math description. The model appears to the users as a "black box". D-PNN combines the neural network functionalities with some math techniques of differential equation solutions. Its models are the boundary of neural network and exact computational techniques.

$$y = a_0 + \sum_{i=1}^{m} a_i x_i + \sum_{i=1}^{m}\sum_{j=1}^{m} a_{ij} x_i x_j + \sum_{i=1}^{m}\sum_{j=1}^{m}\sum_{k=1}^{m} a_{ijk} x_i x_j x_k + \ldots \quad (1)$$

*m – number of variables* $X(x_1, x_2, \ldots, x_m)$ $A(a_1, a_2, \ldots, a_m)$, ... - *vectors of parameters*

D-PNN's block skeleton is formed by the GMDH (Group Method of Data Handling) polynomial neural network, which was created by a Ukrainian scientist Aleksey Ivakhnenko in 1968, when the back-propagation technique was not known yet [4]. General connection between input and output variables is possible to express by the Volterra functional series, a discrete analogue of which is Kolmogorov-Gabor polynomial (1). This polynomial can approximate any stationary random sequence of observations and can be computed by either adaptive methods or system of Gaussian normal equations [6]. GMDH decomposes the complexity of a process into many simpler relationships each described by low order polynomials (2) for every pair of the input values. Typical GMDH network maps a vector input $\boldsymbol{x}$ to a scalar output $y$, which is an estimate of the true function $f(\boldsymbol{x}) = y^t$.

$$y = a_0 + a_1 x_i + a_2 x_j + a_3 x_i x_j + a_4 x_i^2 + a_5 x_j^2 \quad (2)$$

## 2 General Partial Differential Equation Composition

The basic idea of the D-PNN is to compose and substitute a general sum partial differential equation (3), which is not known in advance and can describe a system of dependent variables, with a generated sum of fractional relative multi-parametric polynomial derivative terms (5).

$$a + \sum_{i=1}^{n} b_i \frac{\partial u}{\partial x_i} + \sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij} \frac{\partial^2 u}{\partial x_i \partial x_j} + \ldots = 0 \qquad u = \sum_{k=1}^{\infty} u_k \quad (3),(4)$$

$u = f(x_1, x_2, \ldots, x_n)$ *– searched function of all input variables*
$a$, $B(b_1, b_2, \ldots, b_n)$, $C(c_{11}, c_{12}, \ldots)$ *– polynomial parameters*

Partial DE terms are formed according to the adapted method of integral analogues, which is a part of the similarity model analysis. It replaces mathematical operators and symbols of a DE by ratio of corresponding values. Derivatives are replaced by their integral analogues, i.e. derivative operators are removed and simultaneously with all operators are replaced by similarly or proportion signs in equations to form dimensionless groups of variables [5].

$$u_i = \frac{\left(a_0 + a_1 x_1 + a_2 x_2 + a_3 x_1 x_2 + a_4 x_1^2 + a_5 x_2^2 + \ldots\right)^{m/n}}{b_0 + b_1 x_1 + \ldots} = \frac{\partial^m f(x_1, \ldots, x_n)}{\partial x_1 \partial x_2 \ldots \partial x_m} \tag{5}$$

*n – combination degree of a complete polynomial of n-variables*
*m – combination degree of denominator variables*

The fractional polynomials (5) define partial derivative relations of *n*-input variables. The numerator of a DE term (5) is a polynomial of all *n*-input variables and partly defines an unknown function *u* of eq. (4). The denominator is a derivative part, arose from the partial derivation of the complete *n*-variable polynomial in respect to competent variable(s). The root function of numerator takes the polynomial into competent combination degree to get the dimensionless values [5].

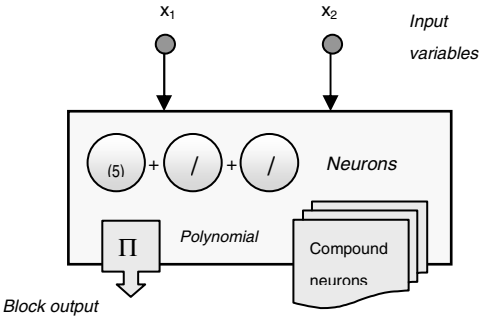



**Fig. 1.** D-PNN block of basic and compound neurons

Blocks of the D-PNN (Fig.1.) consist of derivative neurons, one for each fractional polynomial derivative combination, so each neuron is considered a summation DE term (4). Each block contains a single output polynomial (2), without derivative part. Neurons do not affect the block output but participate direct in the total network output sum calculation of a DE composition. Each block has *1* and neuron *2* vectors of adjustable parameters ***a***, resp. ***a, b***.

$$F\left(x_1, x_2, u, \frac{\partial u}{\partial x_1}, \frac{\partial u}{\partial x_2}, \frac{\partial^2 u}{\partial x_1^2}, \frac{\partial^2 u}{\partial x_1 \partial x_2}, \frac{\partial^2 u}{\partial x_2^2}\right) = 0 \tag{6}$$

*where $F(x_1, x_2, u, p, q, r, s, t)$ is a function of 8 variables*

In the case of 2 input variables the 2nd odder partial DE can be expressed in the form of eq. (6), which involve all derivative terms of variables applied by the GMDH polynomial (2). D-PNN processes these 2-combination square polynomials of blocks and neurons, which form competent DE terms of eq. (5). Each block so include *5* basic neurons of derivatives $x_1$, $x_2$, $x_1x_2$, $x_1^2$, $x_2^2$ of the 2nd order partial DE (6), which is most often used to model physical or natural systems.

## 3 Differential Polynomial Neural Network

Multi-layered networks forms composite polynomial functions (Fig.2.). Compound terms (CT), i.e. derivatives in respect to variables of previous layers, are calculated according to the composite function partial derivation rules (7)(8). They are formed by products of partial derivatives of external and internal functions.

$$F(x_1, x_2, \dots, x_n) = f(y_1, y_2, \dots, y_m) = f(\phi_1(X), \phi_2(X), \dots, \phi_m(X)) \tag{7}$$

$$\frac{\partial F}{\partial x_k} = \sum_{i=1}^{m} \frac{\partial f(y_1, y_2, \dots, y_m)}{\partial y_i} \cdot \frac{\partial \phi_i(X)}{\partial x_k} \qquad k=1, \dots, n \tag{8}$$

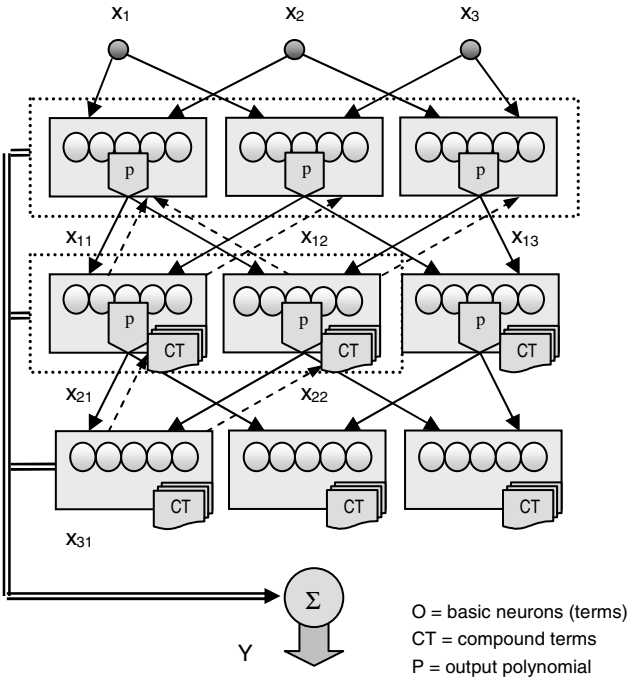



**Fig. 2.** 3-variable multi-layered backward D-PNN with 2-variable combination blocks

Thus blocks of the 2nd and following hidden layers are additionally extended with compound terms (neurons), which form composite derivatives utilizing outputs and inputs of back connected previous layer blocks. The 1st block of the last (3rd) hidden layer (Fig.2.) forms neurons e.g. (9)(10)(11) [10].

$$y_1 = \frac{\partial f(x_{21}, x_{22})}{\partial x_{21}} = w_1 \frac{\left(a_0 + a_1 x_{21} + a_2 x_{22} + a_3 x_{21} x_{22} + a_4 x_{21}^2 + a_5 x_{22}^2\right)^{1/2}}{3/2 \cdot (b_0 + b_1 x_{21})} \tag{9}$$

$$y_2 = \frac{\partial f(x_{21}, x_{22})}{\partial x_{11}} = w_2 \frac{(a_0 + a_1 x_{21} + a_2 x_{22} + a_3 x_{21} x_{22} + a_4 x_{21}^2 + a_5 x_{22}^2)^{1/2}}{3/2 \cdot x_{22}} \cdot \frac{(x_{21})^{1/2}}{3/2 \cdot (b_0 + b_1 x_{11})} \tag{10}$$

$$y_3 = \frac{\partial f(x_{21}, x_{22})}{\partial x_1} = w_3 \frac{(a_0 + a_1 x_{21} + a_2 x_{22} + a_3 x_{21} x_{22} + a_4 x_{21}^2 + a_5 x_{22}^2)^{1/2}}{3/2 \cdot x_{22}} \cdot \frac{(x_{21})^{1/2}}{3/2 \cdot x_{12}} \cdot \frac{(x_{11})^{1/2}}{3/2 \cdot (b_0 + b_1 x_1)} \tag{11}$$

The square (12) and combination (13) derivative terms are also calculated according to the composite function derivation rules.

$$y_4 = \frac{\partial^2 f(x_{21}, x_{22})}{\partial x_{11}^2} = w_4 \frac{(a_0 + a_1 x_{21} + a_2 x_{22} + a_3 x_{21} x_{22} + a_4 x_{21}^2 + a_5 x_{22}^2)^{1/2}}{1.5 \cdot x_{22}} \cdot \frac{x_{21}}{3.8 \cdot (b_0 + b_1 x_{11} + b_2 x_{11}^2)} \tag{12}$$

$$y_5 = \frac{\partial^2 f(x_{21}, x_{22})}{\partial x_{11} \partial x_{12}} = w_5 \frac{(a_0 + a_1 x_{21} + a_2 x_{22} + a_3 x_{21} x_{22} + a_4 x_{21}^2 + a_5 x_{22}^2)^{1/2}}{1.5 \cdot x_{22}} \cdot \frac{x_{21}}{3.3 \cdot (b_0 + b_1 x_{11} + b_2 x_{12} + b_3 x_{11} x_{12})} \tag{13}$$

The best-fit neuron selection is the initial phase of the DE composition, which may apply a proper genetic algorithm (GA). Parameters of polynomials might be adjusted by means of difference evolution algorithm (EA), supplied with sufficient random mutations. The parameter optimization is performed simultaneously with the GA term combination search, which may result in a quantity of local or global error solutions. The number of network hidden layers coincides with a total amount of input variables. There would be welcome to apply an adequate gradient descent method too, which parameter updates result from partial derivatives of polynomial DE terms in respect with the single parameters [7].

$$Y = \frac{\sum_{i=1}^{k} y_i}{k} \qquad k = \textit{amount of active neurons} \tag{14}$$

Only some of all potential combination DE terms (neurons) may participate in the DE composition, in despite of they have an adjustable term weight ($w_i$). D-PNN's total output $Y$ is the sum of all active neuron outputs, divided by their amount $k$ (14). The root mean square error (RMSE) method (15) was applied for the polynomial parameter optimization and neuron combination selection. D-PNN is trained only with a small set of input-output data samples likewise the GMDH algorithm does [6].

$$E = \sqrt{\frac{\sum_{i=1}^{M} \left(y^d - y_i\right)^2}{M}} \rightarrow \min \tag{15}$$

## 4 Test Experiments

The presented 3-variable multi-layered D-PNN (Fig.2.) can be tested to approximate non-linear multi-parametric functions. The D-PNN and ANN models were trained with 24 data samples, randomly generated by benchmark functions from the interval *<10,400>*. The ANN approximation ability falls rapidly outside of training range, while the D-PNN's alternate errors grow just slowly (Fig.3. and Fig.4.). Experiments with other benchmarks (e.g. $x_1+x_2^2+x_3^3$) result in similar outcome graphs. The ANN with 2-hidden layers of neurons applied the sigmoidal activation function and standard back-propagation algorithm. The parameter and weight adjustment of both methods appeared heavy time-consuming and have not succeed any experiment.

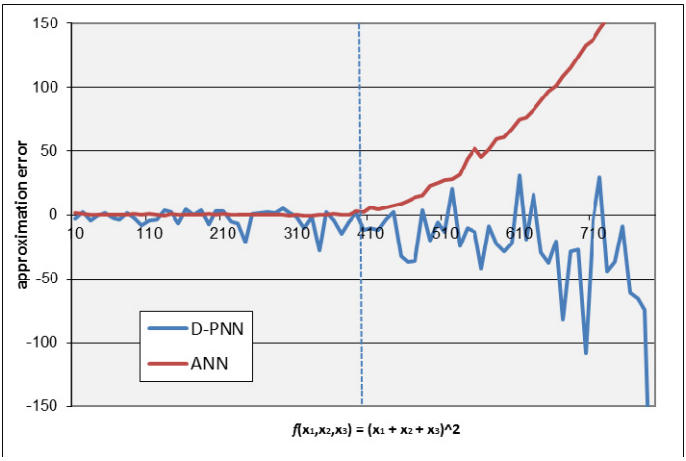



**Fig. 3.** Comparison of the $f(x_1, x_2, x_3)=( x_1+x_2+x_3)^2$ function approximation

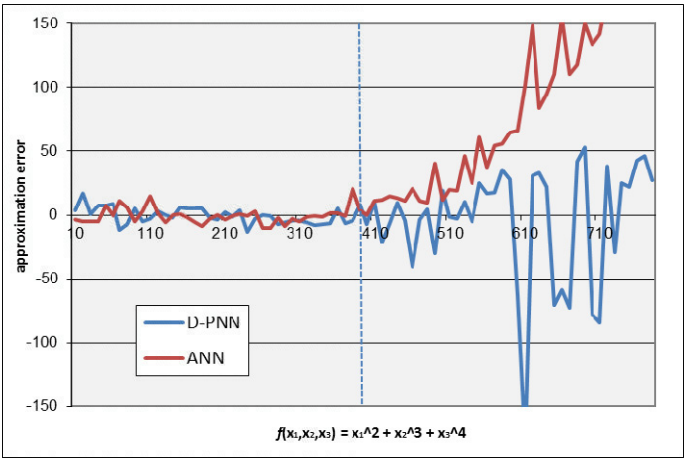



**Fig. 4.** Comparison of the $f(x_1, x_2, x_3)= x_1^2+x_2^3+x_3^4$ function approximation

## 5 Real Data Multi-parametric Models

A real data multi-parametric function can be represented by the relative humidity model which inputs are 3 weather variables of wind speed, temperature and sea level pressure of a 1-site locality. The test model of real meteorological data can roughly estimate the time and amount of precipitations. The relative humidity values increase at night hours (with temperature decrease), upswing or day grows can indicate precipitations (Fig.5a-d). The comparisons were done with 1-layer recurrent neural network (RNN), which applies as inputs also its neuron outputs from a previous time estimate. D-PNN applies only 3 current state variables, which disadvantages it, as it does not allow for time sequences. Both networks were trained with previous day hourly data series (24 or 48 hours, i.e. data samples) free on-line available [11].

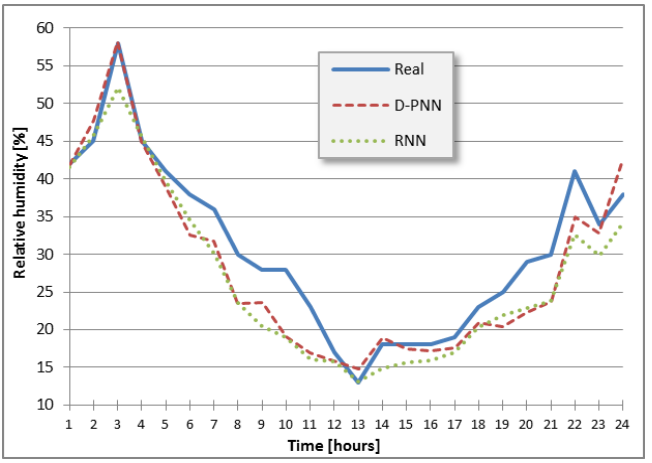



**Fig. 5a.** $RMSE_{D\text{-}PNN} = 4.16$, $RMSE_{RNN} = 4.72$

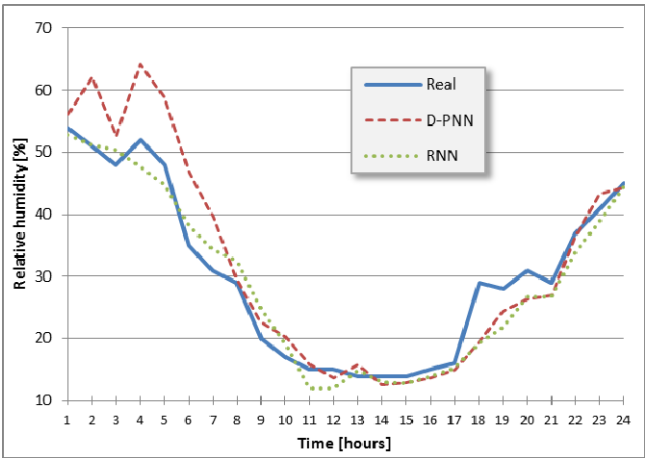



**Fig. 5b.** $RMSE_{D\text{-}PNN} = 5.75$, $RMSE_{RNN} = 3.48$

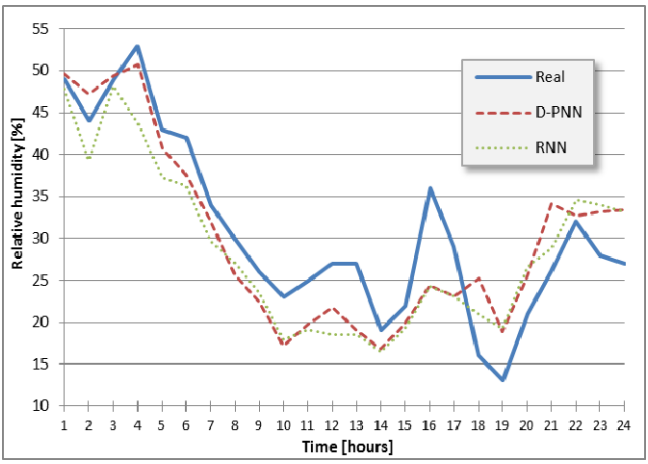



**Fig. 5c.** $RMSE_{D\text{-}PNN} = 5.33$, $RMSE_{RNN} = 5.70$

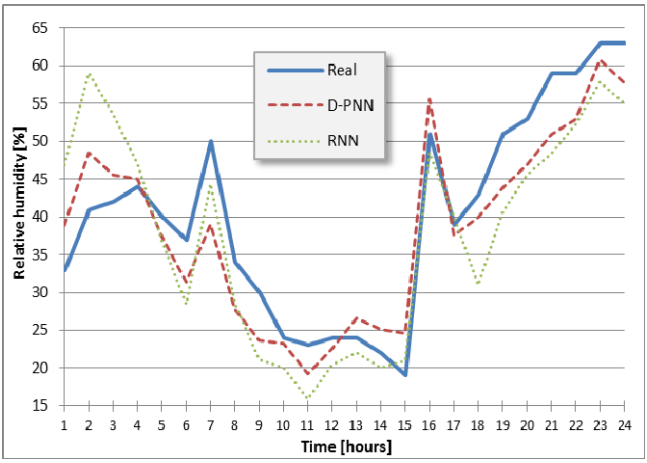



**Fig. 5d.** $RMSE_{D\text{-}PNN} = 5.27$, $RMSE_{RNN} = 7.99$

The humidity value estimation model (Fig.5.) is only a search test of unknown real meteorological data function relations. The model could be formed with reference to other weather variables forecasts, as meteorological predictions of this very complex dynamic system are sophisticated and not any time faithful, using simple neural network models requiring as a rule high amount of input variables. This method could try to improve an official "Aladin" forecast model provided by Czech hydro-meteorological institute. The model comprises 2-day chart prognoses of temperature, humidity, wind speed, pressure and cloudiness in a selected locality [12]. Thus D-PNN can be trained with real data observations of previous 1 or 2 days to define a true multi-parametric function relation, exactly actual for this time interval. After it can form a new revised 24-hour estimation of some variable (e.g. relative humidity),

applying the previous day trained real model of data relations and input variables of the "Aladin" predictions. In the case of an unexpected weather change from a day to day the model will be not equally true, however this trend is not very frequent. The estimation will also depend on prediction accuracies of other "Aladin" model variables, which form the D-PNN's input vector.

## 6 Time-Series Predictions

The simple neural network models applying 3 state weather variables of 1 site locality are able to predict their values simultaneously in this very complex system (Fig.6a-c.). The 3 meteorological variables (wind speed, dew point, see level pressure) and their 3-time series of hour stamps form 9 variables of the input vectors totally [11]. However D-PNN applies only 3 hidden layers of blocks, i.e. 3 inter-connected networks of Fig.2, which disallows it to define all possible data relations.

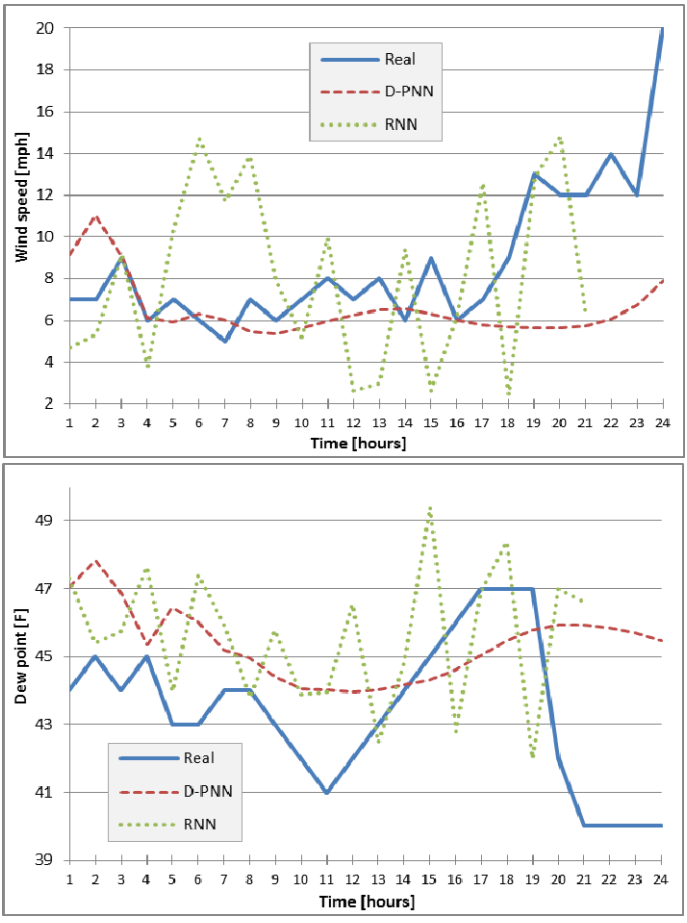
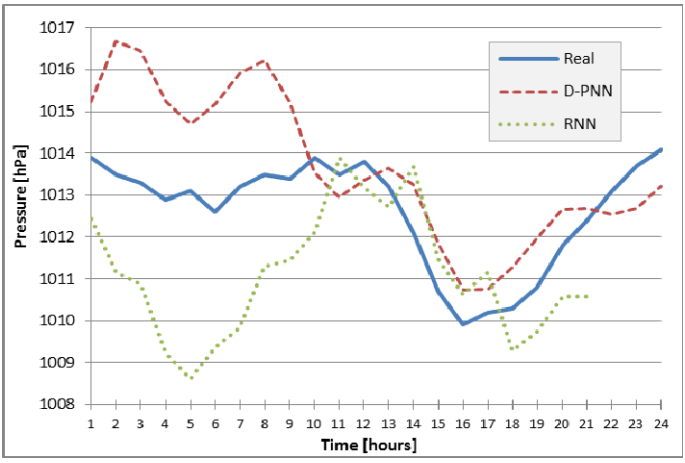



**Fig. 6a-b.** Comparison of weather variables predictions



(c)

**Fig. 6c.** Comparison of weather variables predictions

The D-PNN and RNN (Fig. 7) predict values of the 3-variables at a next time state, which form the 3 latest time inputs of a following prediction. Dew point is the non-linear function of temperature and relative humidity. There was need of doing more time-consuming experiments, which were not always succeeded and also not valid on all tested data as other factors can fair influence the very complex weather system [8].

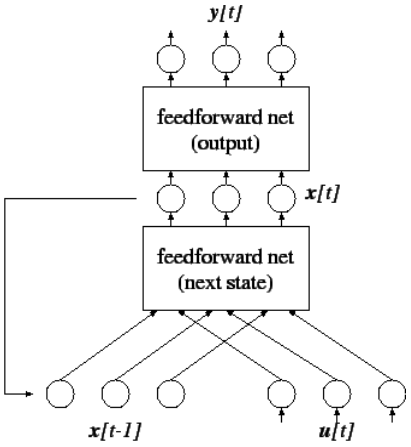



**Fig. 7.** Recurrent neural network

## 7 Conclusion

D-PNN is a new neural network type, which function approximation and identification is based on generalized data relations. Its relative data processing is contrary to the common soft-computing method functionality (e.g. ANN, GP), which

applications are subjected to a fixed interval of absolute values. This handicap disallows it to use far various training and testing data range values, which may involve real data applications. Regarding to test experiments the D-PNN's non-linear regression can cover a wider interval of input values. It forms and resolves an unknown general DE with a composition of sum fractional derivative terms, defining a system model of dependent variables. The inaccuracies of presented experiments can result from applied incomplete rough training and selective methods, requiring improvements. The D-PNN's operating principle differs by far from other common neural network techniques.

**Acknowledgement.** This work has been elaborated in the framework of the IT4Innovations Centre of Excellence project, reg. no. CZ.1.05/1.1.00/02.0070 supported by Operational Programme 'Research and Development for Innovations' funded by Structural Funds of the European Union and by the Ministry of Industry and Trade of the Czech Republic, under the grant no. FR-TI1/420, and by SGS, VŠB – Technical University of Ostrava, Czech Republic, under the grant No. SP2012/58.

## References

1. Dutot, A.L., Rynkiewicz, J., Steiner, F.E., Rude, J.: A 24-h forecast of ozone peaks and exceedance levels using neural classifiers and weather predictions. Environmental Modelling & Software 22 (2007)
2. Giles, C.L.: Noisy Time Series Prediction using Recurrent Neural Networks and Grammatical Inference. Machine Learning 44, 161–183 (2001)
3. Iba, H.: Inference of differential equation models by genetic programming. Information Sciences 178(23), 4453–4468 (2008)
4. Hall, T., Brooks, H.E., Doswell, C.A.: Precipitation Forecasting Using a Neural Network. Weather and Forecasting 14 (1998)
5. Kuneš, J., Vavroch, O., Franta, V.: Essentials of modeling. SNTL Praha (1989) (in Czech)
6. Nikolaev, N.Y., Iba, H.: Adaptive Learning of Polynomial Networks. Springer, New York (2006)
7. Nikolaev, N.Y., Iba, H.: Polynomial harmonic GMDH learning networks for time series modelling. Neural Networks 16, 1527–1540 (2003)
8. Shrivastava, G., Karmakar, S., Kowar, M.K.: Application of Artificial Neural Networks in Weather Forecasting. International Journal of Computer Applications 51(18) (2012)
9. Tsoulos, I., Gavrilis, D., Glavas, E.: Solving differential equations with constructed neural networks. Neurocomputing 72, 2385–2391 (2009)
10. Zjavka, L.: Recognition of Generalized Patterns by a Differential Polynomial Neural Network. Engineering, Technology & Applied Science Research 2(1) (2012)
11. National Climatic Data Center of National Oceanic and Atmospheric Administration (NOAA), `http://cdo.ncdc.noaa.gov/qclcd_ascii/`, `http://cdo.ncdc.noaa.gov/cdo/3505dat.txt`
12. "Aladin" forecast model of Czech hydro-meteorological institute, `http://www.chmi.cz/files/portal/docs/meteo/ov/aladin/results/public/meteogramy/meteogram_page_portal/m.html`

# PSO-2S Optimization Algorithm for Brain MRI Segmentation

Abbas El Dor[1], Julien Lepagnot[2], Amir Nakib[1], and Patrick Siarry[1]

[1] Université de Paris-Est Créteil, Laboratoire LISSI, E.A. 3956
122 rue Paul Armangot, 94400 Vitry-sur-Seine, France
siarry@u-pec.fr
[2] Université de Haute-Alsace, Laboratoire LMIA, E.A. 3993
4 rue des Frères Lumière, 68093 Mulhouse, France

**Abstract.** In image processing, finding the optimal threshold(s) for an image with a multimodal histogram can be done by solving a Gaussian curve fitting problem, *i.e.* fitting a sum of Gaussian probability density functions to the image histogram. This problem can be expressed as a continuous nonlinear optimization problem. The goal of this paper is to show the relevance of using a recently proposed variant of the Particle Swarm Optimization (PSO) algorithm, called PSO-2S, to solve this image thresholding problem. PSO-2S is a multi-swarm PSO algorithm using charged particles in a partitioned search space for continuous optimization problems. The performances of PSO-2S are compared with those of SPSO-07 (*Standard Particle Swarm Optimization in its 2007 version*), using reference images, *i.e.* using test images commonly used in the literature on image segmentation, and test images generated from brain MRI simulations. The experimental results show that PSO-2S produces better results than SPSO-07 and improves significantly the stability of the segmentation method.

## 1 Introduction

Digital image processing has attracted a growing interest, due to its practical relevance in many fields of research and in industrial and medical applications. Image segmentation is typically used to locate objects and boundaries in images. It is one of the main components of several image analysis systems, thus it received a great deal of attention. Several surveys and comparative papers are available in the literature [13,10,14,7]. Image thresholding is one of the most popular segmentation approaches. It makes use of the image histogram to partition the images into several meaningful groups of pixels. In automatic image thresholding methods, the segmentation problem can be formulated as a continuous nonlinear optimization problem. Hence, the use of a metaheuristic is a relevant choice to solve it efficiently.

In this paper, we propose to use a recently proposed algorithm [5], called PSO-2S, which is a new variant of particle swarm optimization (PSO) [8]. PSO is inspired by social behavior simulations of bird flocking. It has already been

J.-S. Pan et al. (eds.), *Genetic and Evolutionary Computing*,
Advances in Intelligent Systems and Computing 238,
DOI: 10.1007/978-3-319-01796-9_2, 

applied successfully to image processing problems [6,9]. This algorithm optimizes a problem by iteratively improving a candidate solution with regard to a given measure of quality. PSO-2S is a multi-swarm PSO algorithm based on several initializations in different zones of the search space, using charged particles. This algorithm uses two kinds of swarms, one main and several auxiliary swarms. The best particles of the auxiliary ones generate the main swarm. More precisely, the auxiliary swarms are initialized several times in different zones. An electrostatic repulsion heuristic is then applied in each zone to increase the diversity of the particles. Each auxiliary swarm performs several generations based on standard PSO algorithm to provide the best solution in its related zone. The provided solutions are then used as the main swarm.

This paper is structured as follows: Section 2 presents an overview of the standard particle swarm optimization and its new variant PSO-2S. Section 3 is dedicated to the presentation of the image thresholding method. The image segmentation criterion is given in Section 4. Experimental protocol and parameter setting are presented in Section 5. Experimental results are discussed in Section 6. The work in this paper is concluded in section 7.

## 2 Presentation of the PSO-2S Algorithm

### 2.1 Review of the Standard PSO

The particle swarm optimization (PSO) [8] is inspired originally by the social and cognitive behavior existing in the bird flocking. The algorithm is initialized with a population of particles randomly distributed in the search space, and each particle is assigned a randomized velocity. Each particle represents a potential solution to the problem.

In this paper, the swarm size is denoted by $s$, and the search space is n-dimensional. In general, the particles have three attributes: the current position $X_i = (x_{i,1}, x_{i,2}, ..., x_{i,n})$, the current velocity vector $V_i = (v_{i,1}, v_{i,2}, ..., v_{i,n})$ and the past best position $Pbest_i = (p_{i,1}, p_{i,2}, ..., p_{i,n})$. The best position found in the neighborhood of the particle $i$ is denoted by $Gbest_i = (g_1, g_2, ..., g_n)$. These attributes are used to update iteratively the state of each particle in the swarm. The objective function to be minimized is denoted by $f$. The velocity vector $V_i$ of each particle is updated using the best position it visited so far and the overall best position visited by its neighbors. Then, the position of each particle is updated using its updated velocity per iteration. At each step, the velocity of each particle and its new position are updated as follows:

$$v_{i,j}(t+1) = wv_{i,j}(t) + c_1 r_{1_{i,j}}(t)\,[pbest_{i,j}(t) - x_{i,j}(t)] + c_2 r_{2_{i,j}}(t)\,[gbest_{i,j}(t) - x_{i,j}(t)] \quad (1)$$

$$x_{i,j}(t+1) = x_{i,j}(t) + v_{i,j}(t+1) \quad (2)$$

where $w$ is called inertia weight, $c_1, c_2$ are the learning factors and $r_1, r_2$ are two random numbers selected uniformly in the range $[0, 1]$.

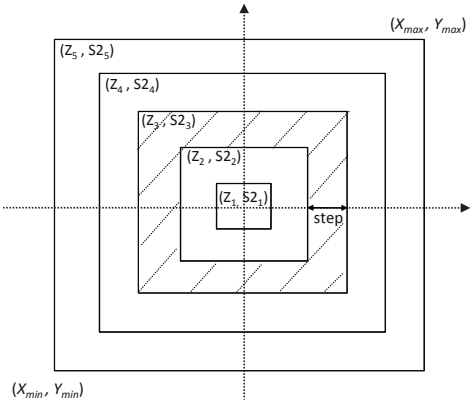



**Fig. 1.** Partitioning of the search space

### 2.2 PSO Improved Variant: PSO-2S

An improved variant of the original PSO algorithm, called PSO-2S, was proposed by El Dor et al. [5]. This variant consists of using three main ideas: the first is to use two kinds of swarms: a main swarm, denoted by S1, and $s$ auxiliary ones, denoted by $S2_i$, where $1 \leq i \leq s$. The second idea is to partition the search space into several zones in which the auxiliary swarms are initialized (the number of zones is equal to the number of auxiliary swarms $s$). The last idea is to use the concept of the electrostatic repulsion heuristic to diversify the particles for each auxiliary swarm in each zone.

To construct S1, the auxiliary swarms $S2_i$ evolve several times in different areas, and then each best particle for each $S2_i$ is saved and considered as a new particle of S1. To do so, the population of each auxiliary swarm is initialized randomly in different zones (each $S2_i$ is initialized in its corresponding zone $i$). After each of these initializations, $nb_{generation}$ displacements of particles, for each $S2_i$, are performed in the same way as standard PSO. Then the best solution found by each auxiliary swarm, named $gbest_i$, is added to S1. The number of initializations of $S2_i$ is equal to the number of particles in S1.

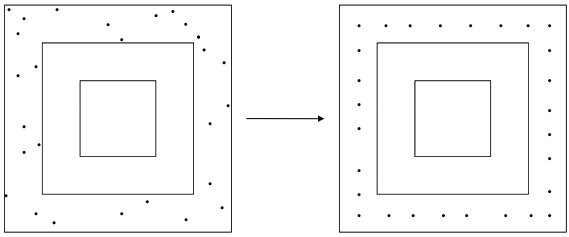
As mentioned above, the second idea is to partition the search space $[min_d, max_d]^D$ into several zones ($max_{zone}$ zones). Then, one calculates the $center_d$ and the $step_d$ of each dimension separately, according to (3) and (4). In the case of using an uniform (square) search space, the $step_d$ are similar for all dimensions.

$$center_d = (max_d + min_d)/2 \tag{3}$$

$$step_d = (max_d - min_d)/2 \times max_{zone} \tag{4}$$

where $max_{zone}$ is a fixed value, and $d$ is the current dimension ($1 \leq d \leq D$).

This process is illustrated in Figure 1, where the $i^{th}$ swarm $S2_i$ and its attributed zone $Z_i$ are denoted by ($Z_i$, $S2_i$).



(a) (b)

**Fig. 2.** Repulsion process: (a) ($Z_3$, $S2_3$) before repulsion, (b) ($Z_3$, $S2_3$) after repulsion

The sizes of the zones of the partitioned search space are different ($Z_1 < Z_2 < \ldots < Z_{max_{zone}}$). Therefore, the number of particles in $S2_i$, denoted by $S2_{isize}$, depends on its corresponding zone size. Indeed, a small zone takes less particles and the number of particles increases when the zone becomes larger. The size of each auxiliary swarm is calculated as follows:

$$S2_{isize} = num_{zone} \times nb_{particle} \tag{5}$$

where $num_{zone} = 1, 2, ..., max_{zone}$ is the current zone number and $nb_{particle}$ is a fixed value.

After the initializations of the auxiliary swarms in different zones ($Z_i$, $S2_i$), an electrostatic repulsion heuristic is applied to diversify the particles and to widely cover the search space [4]. This technique is used in an agent-based optimization algorithm for dynamic environments [11]. Therefore, this procedure is applied in each zone separately, hence each particle is considered as an electron. Then a force of $1/r^2$ is applied, on the particles of each zone, until the maximum displacement of a particle during an iteration becomes lower than a given threshold $\epsilon$ (where $r$ is the distance between two particles, $\epsilon$ is typically equal to $10^{-4}$). At each iteration of this procedure, the particles are projected in the middle of the current zone, before a new application of the repulsion heuristic. Figure 2 presents an example of the repulsion applied to ($Z_3$, $S2_3$).

## 3 The Problem at Hand

The segmentation problem has received a great deal of attention, thus any attempt to survey the literature would be too space-consuming. The most popular segmentation methods may be found in [15]. In this work, image segmentation is performed using the thresholding approach. Image thresholding is a supervised segmentation method, *i.e.* the number of regions (classes of pixels) and their properties are known in advance by the user. The segmentation is done by determining, for each pixel, the class whose properties are the closest to those observed for that pixel. The thresholding technique is based on the assumption that different regions of the image can be distinguished by their gray levels.

It makes use of the histogram $h(j)$ of the processed image, *i.e.* the observed probability of gray level $j$. It can be defined as follows:

$$h(j) = \frac{g\,(j)}{\sum_{i=0}^{L-1} g\,(i)} \tag{6}$$

where $g(j)$ denotes the occurrence of gray-level $j \in \{0, 1, \ldots, L-1\}$ in the image.

Thresholding the image into $N$ classes is to find the $N-1$ thresholds that will partition the histogram into $N$ zones.

The main contribution of the work we present here is to show the significance of using PSO-2S for MR image segmentation. The performances of PSO-2S are first compared with those of SPSO-07 (*Standard Particle Swarm Optimization* in its 2007 version) [2], using reference images, commonly used in the literature on image segmentation. Then, the performances of both algorithms are compared using images from a database generated by brain MRI simulation [1,3]. This database, called *BrainWeb*, provides images for which an "optimal" segmentation is known. Indeed, the BrainWeb MRI simulations are based on a predefined anatomical model of the brain. The images generated by these simulations can then be used to validate a segmentation method, or to compare the performance of different methods.

## 4 Image Segmentation Criterion

Before using this criterion we must fit the histogram of the image to be segmented to a sum of Gaussian probability density functions (pdf's). This procedure is named Gaussian curve fitting, more details about it are given below. The pdf model must be fitted to the image histogram, typically by using the maximum likelihood or mean-squared error approach, in order to find the optimal threshold(s). For the multimodal histogram $h(i)$ of an image, where $i$ is the gray level, we fit $h(i)$ to a sum of $d$ probability density functions [12]. The case where the Gaussian pdf's are used is defined by:

$$p\,(x) = \sum_{i=1}^{d} P_i \exp\left[-\frac{(x-\mu_i)^2}{\sigma_i^2}\right] \tag{7}$$

where $P_i$ is the amplitude of Gaussian pdf on $\mu_i$, $\mu_i$ is the mean and $\sigma_i^2$ is the variance of mode $i$, and $d$ is the number of Gaussians used to approximate the original histogram and corresponds to the number of segmentation classes.

Our goal is to find a vector of parameters, $\Theta$, that minimizes the fitting error $J$, given by the following expression:

$$J(\Theta) = \sum_{i=0}^{L-1} \left|h(i) - p(\Theta, i)\right|^2 \tag{8}$$

where $h(i)$ is the measured histogram. Here, $J$ is the objective function to be minimized with respect to $\Theta$, a set of parameters defining the Gaussian pdf's

and the probabilities, given by $\Theta = \{P_i, \quad \mu_i, \quad \sigma_i; \quad i = 1, 2, \cdots, d\}$. After fitting the multimodal histogram, the optimal threshold could be determined by minimizing the overall probability of error, for two adjacent Gaussian pdf's, given by:

$$e(T_i) = P_i \int_{-\infty}^{T_i} p_i(x)\, dx + P_{i+1} \int_{T_i+1}^{\infty} p_{i+1}(x)\, dx \tag{9}$$

with respect to the threshold $T_i$, where $p_i(x)$ is the $i^{th}$ pdf and $i = 1, \ldots, d-1$. Then the overall probability to minimize is:

$$E(T) = \sum_{i=1}^{d-1} e(T_i) \tag{10}$$

where $T$ is the vector of thresholds: $0 < T_1 < T_2 < ... < T_{(d-1)} < L-1$. In our case $L$ is equal to 256.

## 5 Experimental Protocol and Parameter Setting

To compare the performance of PSO-2S and SPSO-07, the criterion (8) is minimized for each test image in Figure 3 (a). The stagnation criterion used is satisfied if no significant improvement (greater than 1E−10) in the current best solution is observed during 1E+4 successive evaluations of the objective function. In addition, the maximum number of evaluations allowed is set to 300000.

In this figure, LENA and BRIDGE are reference images used for the validation of segmentation methods in the literature. The images MRITS and MRICS are obtained from BrainWeb [1] and correspond to transverse and coronal sections of a brain, respectively. The parameters used for the MRI simulation are a T1-weighted sequence, a slice thickness of 1mm, a Gaussian noise of 3% calculated relative to the brightest tissue, and a 20% level of intensity non-uniformity (radio frequency bias).

The values of the PSO and SPSO-07 parameters used for the segmentation problem are defined below:

- PSO-2S using 30 zones and $\frac{p^4}{7000} + 10$ particles in each zone, where $p$ is the zone number. The parameter $K$ used to generate the neighborhood of the particles is set to $K = 3$. The parameter $Nb_{generation}$ is set to 15 ;
- SPSO-07 (*Standard Particle Swarm Optimization* in its 2007 version) [2] using $10+2\sqrt{D}$ particles (the formula recommended by the authors of SPSO-07), where $D$ is the dimension of the problem. The parameter $K$ is set to $K = 3$.

## 6 Experimental Results and Discussion

In this section, the experimental results obtained with PSO-2S and SPSO-07 are presented. The segmentation results are shown in Figure 3. In this figure, the

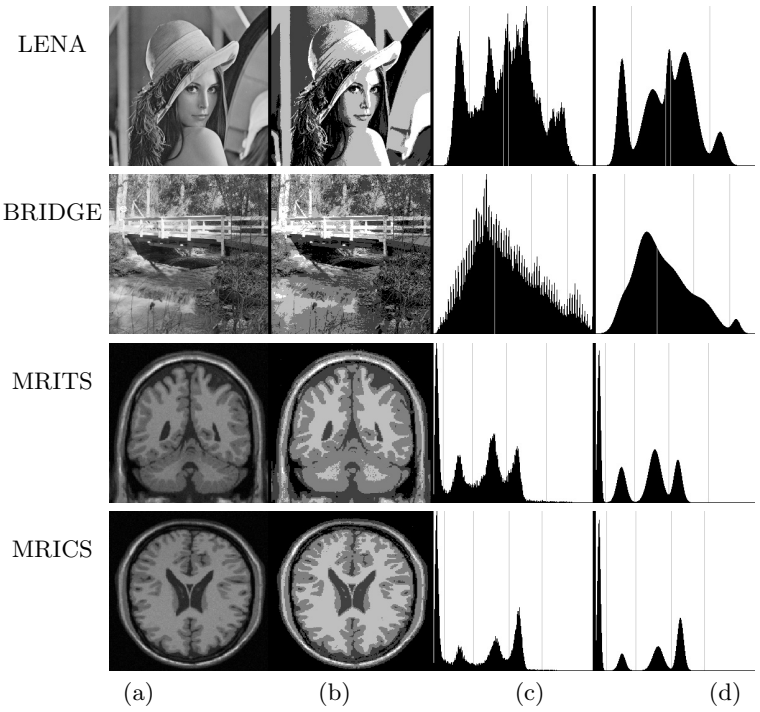



**Fig. 3.** Illustration of the segmentation process. (a) Original images. (b) Segmented images using thresholds in Table II. (c) Original histograms. (d) Approximated histograms.

original images and their histograms are illustrated in (a) and (c), respectively. Approximated histograms are presented in (d), and segmented images (using 5 classes) are shown in (b). For each test image, one can see that the approximation of its histogram, illustrated in detail for LENA in Figure 4, leads to a good image segmentation.

The histogram approximation results, for each test image, are presented in Table 1. In this table, the parameters of each of the five Gaussian pdf's used to approximate the histogram of each image are given. The parameters of the $i^{\text{th}}$ pdf of an image are denoted by $P_i$, $\mu_i$ and $\sigma_i$. Threshold values between the different classes of pixels, calculated for each image using its approximated histogram, are given in Table 2.

For each test image, the number of evaluations performed by each algorithm, averaged over 100 runs, is given in Table 3. The success rate (the percentage of acceptable solutions found among the ones of the 100 runs, *i.e.* the percentage of solutions with an objective function value lower or equal to 5.14E−4, 5.09E−4, 7.51E−4, 7.99E−4 for LENA, BRIDGE, MRITS, MRICS, respectively), and the average approximation error (the average value of the objective function for the

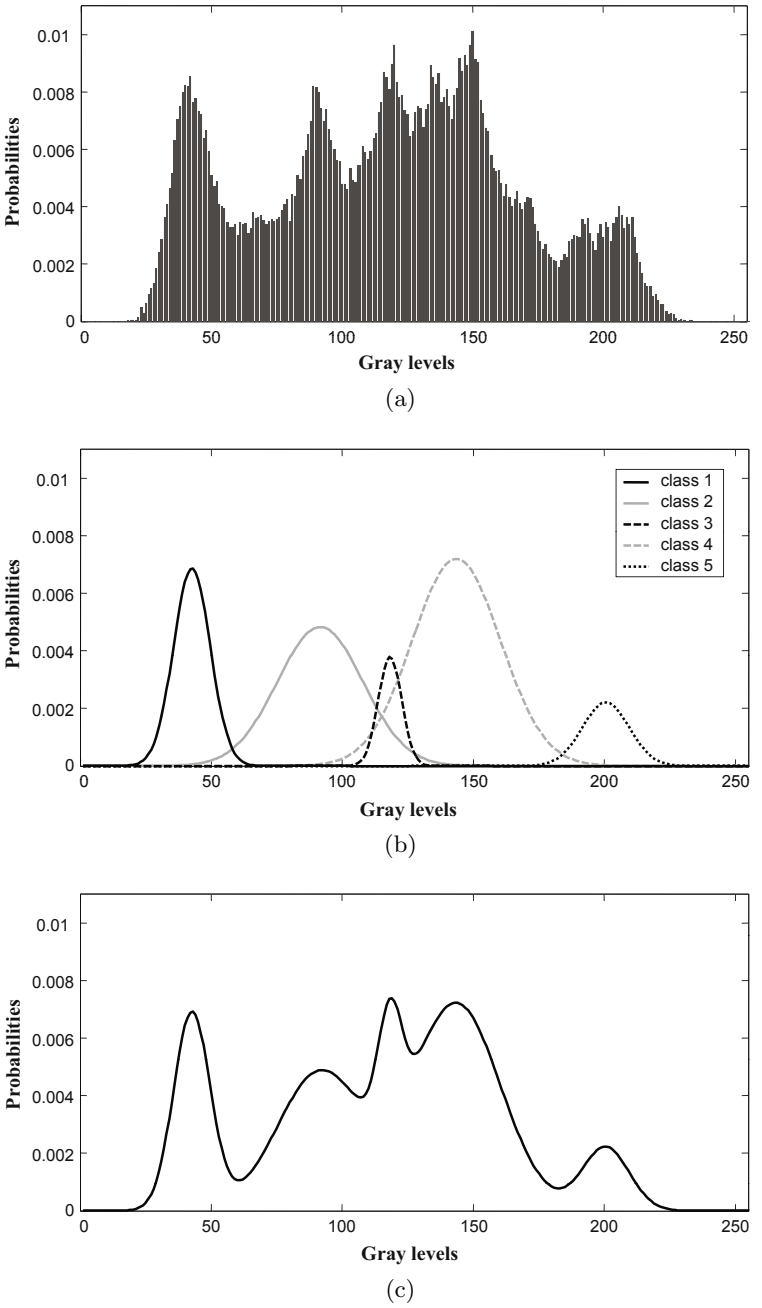



**Fig. 4.** Illustration of the histogram approximation process for LENA. (a) Original histogram. (b) Gaussian pdf's for each class of pixels. (c) Sum of the Gaussian pdf's (approximated histogram).

**Table 1.** Parameters of the Gaussian pdf's used to approximate the histogram of each test image

| Image | $\mu_1$ | $P_1$ | $\sigma_1$ | $\mu_2$ | $P_2$ | $\sigma_2$ | $\mu_3$ | $P_3$ | $\sigma_3$ | $\mu_4$ | $P_4$ | $\sigma_4$ | $\mu_5$ | $P_5$ | $\sigma_5$ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| LENA | 41.62 | 0.17 | 9.87 | 90.57 | 0.28 | 23.14 | 117.21 | 0.06 | 6.33 | 142.54 | 0.43 | 23.82 | 199.56 | 0.07 | 12.64 |
| BRIDGE | 41.13 | 0.06 | 15.36 | 76.75 | 0.39 | 25.97 | 119.25 | 0.38 | 34.73 | 173.38 | 0.14 | 28.43 | 225.53 | 0.02 | 9.09 |
| MRITS | 4.71 | 0.28 | 3.78 | 40.79 | 0.17 | 10.05 | 94.50 | 0.36 | 14.26 | 131.55 | 0.19 | 9.17 | 225.60 | 0.00 | 252.92 |
| MRICS | 4.66 | 0.41 | 3.66 | 41.76 | 0.09 | 7.64 | 99.57 | 0.21 | 11.92 | 135.53 | 0.29 | 7.46 | 192.73 | 0.00 | 239.82 |

**Table 2.** Threshold values for each test image

| Image | Thresholds |
|---|---|
| LENA | 57 112 120 183 |
| BRIDGE | 46 98 156 215 |
| MRITS | 15 62 117 184 |
| MRICS | 17 64 121 177 |

**Table 3.** Average number of evaluations for the segmentation of an image, approximation error and success rate obtained for each algorithm, for each test image

| Image | Algorithm | Evaluations | Approximation error | Success rate |
|---|---|---|---|---|
| LENA | PSO2S | 119721.7 ± 64844.3 | 5.44E−4 ± 3.06E−5 | 41 % |
| | SPSO-07 | 67057.1 ± 64316.9 | 5.52E−4 ± 3.08E−5 | 25 % |
| BRIDGE | PSO2S | 241537.8 ± 70707.8 | 5.27E−4 ± 9.54E−5 | 48 % |
| | SPSO-07 | 125524.4 ± 63277.9 | 5.16E−4 ± 6.01E−6 | 24 % |
| MRITS | PSO2S | 81080.9 ± 66569.8 | 7.69E−4 ± 1.26E−4 | 95 % |
| | SPSO-07 | 44212.0 ± 57321.5 | 8.79E−4 ± 2.73E−4 | 77 % |
| MRICS | PSO2S | 72922.5 ± 35894.4 | 8.68E−4 ± 1.18E−4 | 70 % |
| | SPSO-07 | 28185.4 ± 16188.4 | 9.46E−4 ± 2.71E−4 | 54 % |

best solution found) of an image histogram are also given in this table, for 100 runs of an algorithm.

In this table, we see that PSO-2S requires more evaluations than SPSO-07 to converge to an acceptable solution. However, its success rate is significantly higher than the one of SPSO-07 for all images, according to the Fisher's exact test with a 95% confidence level. Indeed, PSO-2S is designed to prevent premature convergence of PSO algorithm. Hence, it significantly improves the stability of the segmentation method. It shows the significance of using PSO-2S for this class of problems.

## 7 Conclusion

In this paper, we present an image segmentation method using the thresholding approach to identify several classes of pixels in standard and medical images. This method includes an optimization step in which we integrated our PSO-2S algorithm. We also tested the method using the algorithm SPSO-07.

Segmentation results obtained on several test images, commonly used in the literature in image processing and on synthetic images obtained from simulations of brain MRI, are satisfactory. We show that using PSO-2S provides greater stability for this segmentation method, compared with SPSO-07. It shows the relevance of using PSO-2S for this type of problems. Our work in progress consists in the improvement of the segmentation criterion in order to enhance the segmentation quality and accelerate the optimization process.

## References

1. BrainWeb: Simulated Brain Database, `http://brainweb.bic.mni.mcgill.ca/brainweb` (2012)
2. Clerc, M., et al.: The Particle Swarm Central website (2012), `http://www.particleswarm.info`
3. Collins, D.L., Zijdenbos, A.P., Kollokian, V., Sled, J.G., Kabani, N.J., Holmes, C.J., Evans, A.C.: Design and construction of a realistic digital brain phantom. IEEE Transactions on Medical Imaging 17(3), 463–468 (1998)
4. Conway, J., Sloane, N.: Sphere Packings, Lattices and Groups. Springer (1999)
5. El Dor, A., Clerc, M., Siarry, P.: A multi-swarm PSO using charged particles in a partitioned search space for continuous optimization. Computational Optimization and Applications 53(1), 271–295 (2012)
6. Feng, H.-M., Horng, J.-H., Jou, S.-M.: Bacterial Foraging Particle Swarm Optimization Algorithm Based Fuzzy-VQ Compression Systems. Journal of Information Hiding and Multimedia Signal Processing 3(3), 227–239 (2012)
7. Gonzalez, R.C., Woods, R.E.: Digital Image Processing, 3rd edn. Prentice-Hall, Inc., Upper Saddle River (2006)
8. Kennedy, J., Eberhart, R.C.: Particle swarm optimization. In: The IEEE International Conference on Neural Networks IV, Perth, Australia, November 27-December 1, pp. 1942–1948 (1995)
9. Kwok, N.M., Wang, D., Ha, Q.P., Fang, G., Chen, S.Y.: Locally-Equalized Image Contrast Enhancement Using PSO-Tuned Sectorized Equalization. In: Chatterjee, A., Siarry, P. (eds.) Computational Intelligence in Image Processing, pp. 21–36. Springer (2013)
10. Lee, S.U., Chung, S.Y., Park, R.H.: A comparative performance study of several global thresholding techniques for segmentation. Computer Vision, Graphics, and Image Processing 52(2), 171–190 (1990)
11. Lepagnot, J., Nakib, A., Oulhadj, H., Siarry, P.: A new multiagent algorithm for dynamic continuous optimization. International Journal of Applied Metaheuristic Computing 1(1), 16–38 (2010)
12. Nakib, A., Oulhadj, H., Siarry, P.: Non-supervised image segmentation based on multiobjective optimization. Pattern Recognition Letters 29(2), 161–172 (2008)
13. Pitas, I.: Digital Image Processing Algorithms and Applications. John Wiley & Sons (2000)
14. Sahoo, P.K., Soltani, S., Wong, A.K.C., Chen, Y.C.: A survey of thresholding techniques. Comput. Vision Graph. Image Process. 41(2), 233–260 (1988)
15. Sezgin, M., Sankur, B.: Survey over image thresholding techniques and quantitative performance evaluation. Journal of Electronic Imaging 13, 146–165 (2004)

# Task Scheduling in Grid Computing Environments

Yi-Syuan Jiang and Wei-Mei Chen*

Department of Electronic and Computer Engineering,
National Taiwan University of Science and Technology,
Taipei 106, Taiwan
{M9902144,wmchen}@mail.ntust.edu.tw

**Abstract.** A grid computing environment is a parallel and distributed system that brings together various computing capacities to solve large computation problems. Task scheduling is a critical issue for grid computing, which maps tasks onto a parallel and distributed system for achieving good performance in terms of minimizing the overall execution time. This paper presents a genetic algorithm to solve this problem for improving the existing genetic algorithm with two main ideas: a new initialization strategy is introduced to generate the first population of chromosomes and the good characteristics of found solutions are preserved for new generations. Our proposed algorithm is implemented and evaluated using a set of well-known applications in our specific-defined system environment. The experimental results show that the proposed algorithm outperforms other algorithms within several parameter settings.

## 1 Introduction

In past few years, grid computing systems and applications become popular [3], due to a rapid development of many-core. A grid computing environment is a parallel and distributed system that brings together various computing capacities to solve large computation problems. In grid environments, task scheduling, which plays an important role, divides a larger job into smaller tasks and maps tasks onto a parallel and distributed system [1,6]. The goal of a task scheduling is typically to schedule all the tasks on a given number of available processors so as to minimize the overall length of time required to execute the whole program.

A parallel and distributed computing system may be homogeneous [10] or heterogeneous systems [7,11,13]. A homogeneous system means that the processors are the same performance in processing capabilities. On the other hand, heterogeneous systems have different processing capabilities in the target system. In general, the processors are connected by an interconnection network, which is either fully-connected [11,13] or partially-connected [2]. In the fully-connected network every processor can communicate with each other, whereas data can be transferred to some specified processors in a partially-connected network. Besides, the task duplication issue [10] was also discussed to reduce the communication time by duplicating some tasks on more than one processor to eliminate communication cost. To avoid increasing energy consumption,

* This work is partially supported by National Science Council under the Grant NSC 101-2221-E-011-039-MY2.

J.-S. Pan et al. (eds.), *Genetic and Evolutionary Computing,*
Advances in Intelligent Systems and Computing 238,
DOI: 10.1007/978-3-319-01796-9_3, 

here we consider the target system which is the fully-connected heterogeneous systems without task duplication.

The genetic algorithm (GA), first proposed by Holland [5], provides a popular solution for application problems [4,9]. GAs have been shown that outperforms several algorithms in the task scheduling problem, which simply define the search space to be the solution space in which each point is denoted by a number string, called a chromosome. Based on these solutions, three operators which are selection, crossover, and mutation, are employed to transform a population of chromosomes to better solutions iteratively. In order to keep the good features from the previous generation, the crossover operator exchanges the information from two chromosomes chosen randomly, and the mutation operator alters one bit of a chromosome.

In this paper, we proposed a genetic algorithm for task scheduling on a grid computing system, called TSGA. In general, GA approaches directly initialize the first population by some uniform random process. TSGA develops a new initialization policy, which divides the search space into specific patterns in order to accelerate the convergence of solutions. To solve the task scheduling problem, a chromosome usually contains a mapping part and an order part to indicate the corresponding computer and the executing order. In the standard GA, when crossover and mutation operators are applied, both of the mapping part and the order part will be changed, which brings that the parents' characteristics cannot be kept in the next generation. Inspired by the idea of eugenics, TSGA presents new operators for crossover and mutation to preserve good features from the previous generation.

The remainder of the paper is organized as follow. In the next section, we provide the problem definition. The proposed genetic scheduling algorithm is presented in Section 3. We describe our experimental results in Section 4. Finally, conclusions are drawn in Section 5.

## 2 Problem Definition

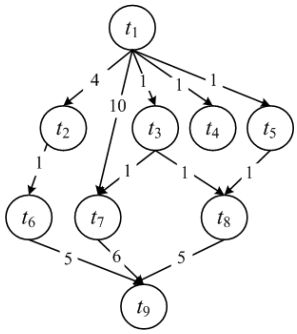
Task scheduling is mapping smaller tasks to multiprocessors. Tasks with data precedence are modeled by a Directed Acyclic Graph (DAG) [13]. The main idea of DAG scheduling is minimizing the makespan which is the overall execution time for all tasks.

### 2.1 DAG Modeling

A DAG $G = (V,E)$ is depicted in Fig. 1(a), where $V$ is a set of $N$ nodes and $E$ is a set of $M$ directed edges. For the problem of task scheduling, $V$ represents the set of tasks and each task contains a sequence of instructions that should be completed in a particular order. Let $w_{i,j}$ be the computation time to finish a particular task $t_i \in V$ on the processor $P_j$, detailed in Fig. 1(b). Each edge $e_{i,j} \in E$ in the DAG indicates the precedence constraint that task $t_i$ should complete its execution before task $t_j$ starts. Let $c_{i,j}$ denote the communication cost needed to transport the data between task $t_i$ and task $t_j$, which is the weight on an edge $e_{i,j}$. If $t_i$ and $t_j$ is assigned to the same processor, the communication cost $c_{i,j}$ is zero.

The source node of an edge is called a predecessor of that node. Similarly, the destination node emerged from a node is called a successor of that node. In Fig. 1(a), $t_1$ is

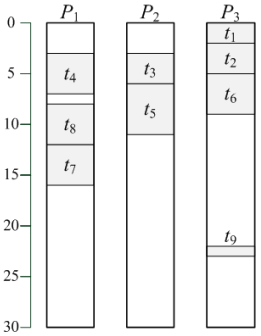
the predecessor of $t_2$, $t_3$, $t_4$, and $t_5$. On the other hand, $t_2$, $t_3$, $t_4$, and $t_5$ are the successor of $t_1$. In a graph, a node with no parent is called an entry node, and a node with no child is called an exit node. If a node $t_i$ is scheduled to a processor $P_j$, the start-time and the finish-time of $t_i$ are denoted by $ST(t_i,P_j)$ and $FT(t_i,P_j)$, respectively.




(a)

| task | $P_1$ | $P_2$ | $P_3$ |
|---|---|---|---|
| 1 | 3 | 1 | 2 |
| 2 | 3 | 3 | 3 |
| 3 | 4 | 3 | 2 |
| 4 | 4 | 2 | 6 |
| 5 | 7 | 5 | 3 |
| 6 | 2 | 6 | 4 |
| 7 | 4 | 6 | 2 |
| 8 | 4 | 3 | 5 |
| 9 | 1 | 1 | 1 |

(b)




(c)

**Fig. 1.** An example of (a) DAG, (b) the computation cost matrix, and (c) an example of scheduling

## 2.2 Makespan

After all tasks are scheduled onto parallel processors, considering a particular task $t_i$ on the processor $P_j$, the start-time $ST(t_i,P_j)$ can be defined as

$$ST(t_i,P_j) = \max\{RT_j, DAT(t_i,P_j)\},$$

where $DAT(t_i,P_j)$ is the data arrival time of task $t_i$ at the processor $P_j$, which is the time when all the needed data have been transmitted. On the other hand, $DAT(t_i,P_j)$ is defined as

$$DAT(t_i,P_j) = \max_{t_k \in pred(t_i)} \{(FT(t_k,P_j) + c_{k,i})\},$$

where $pred(t_i)$ denotes the set of immediate predecessor tasks of the task $t_i$. Since

$$FT(t_k,P_j) = w_{k,j} + ST(t_k,P_j)$$

and

$$RT_j = \max_{t_k \in exe(P_j)} \{FT(t_k,P_j)\},$$

where $exe(P_j)$ is the set containing tasks which executes on the processor $P_j$, the overall schedule length of the entire program is the largest finish time among all tasks and can be expressed as

$$makespan = \max_{t_i \in V} \{FT(t_i,P_j)\}.$$

Fig. 1(c) demonstrates a scheduling for the graph described in Fig. 1(a). The makespan of this scheduling is 23.

## 3 Proposed Method

In this section, we introduce TSGA algorithm in detail, including the encoded and decoded representations and five important operators.

### 3.1 The Representation of Solutions

The representation of a chromosome is given in Fig. 2, which is divided into a mapping part ($S_M$) and an order part ($S_O$). We use integer arrays to store $S_M$ and $S_O$ and the size of arrays is equal to the number of tasks. If $S_M[i]$ is $j$ and $S_O[i]$ is $k$, it means that a task $t_k$ is executed on the processor $P_j$.

According to the chromosome represented in Fig. 2, the solution of a DAG in Fig. 1(a) can be scheduled in Fig. 1(c). First, we assign tasks into the mapping processor according to the index of $S_M$. Tasks $t_4$, $t_7$, and $t_8$ are scheduled on processor $P_1$. Tasks $t_3$, and $t_5$ are executed on processor $P_2$. Tasks $t_1$, $t_2$, $t_6$, and $t_9$ are assigned to the processor $P_3$. Following the order in $S_O$, we schedule $t_4$, $t_7$, and $t_8$ in the order of $t_4$, $t_8$, $t_7$ in $P_1$. For $P_2$, $t_3$ is executed before $t_5$. Tasks $t_1$, $t_2$, $t_6$, and $t_9$ are taken in the order of $t_1$, $t_2$, $t_6$, $t_9$ in $P_3$. Finally, we should count the wait time for communicating, if two dependent tasks are scheduled on a different processor.

TSGA defines the fitness function in order to measure the quality of solutions. The purpose of the scheduling problem is minimizing the makespan. Thus, the fitness function is defined as the makespan.

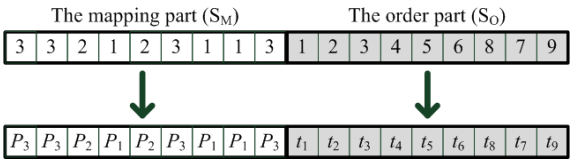



**Fig. 2.** A representation of chromosome

### 3.2 TSGA

An algorithmic flowchart of TSGA is given in Fig. 3. If the number of generation is not smaller than the maximum generation $G$, TSGA will output the best solution.

**Initialization.** As shown in Algorithm 1, TSGA initializes the first population that consists of encoded chromosomes. Each chromosome is composed of the processor assignment and the execution order. Instead of using the random strategy to give the processor assignment, we devise a new method dividing the search space into specific patterns equally. The search space is divided into $\log_2 n$ subspaces, where $n$ is the number of processors. Since the best scheduling solution may occupy either few processors or most processors in different cases, we give some patterns with a different number of processors in order to explore the solution space in different aspects.

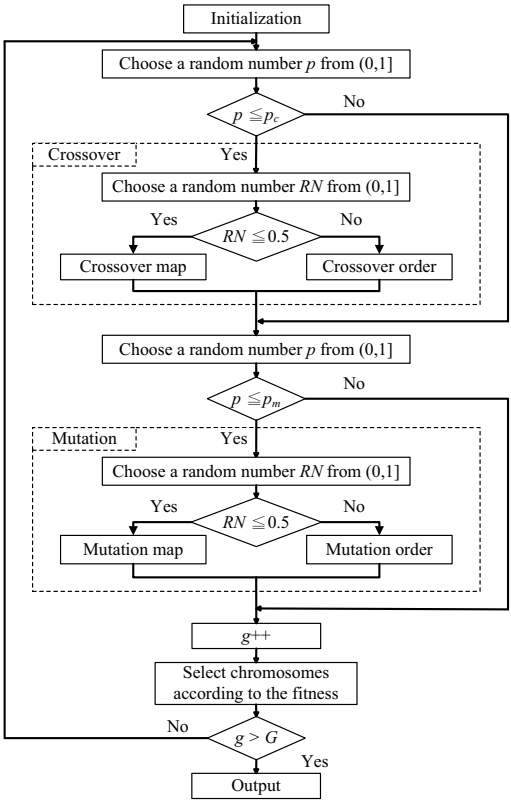



**Fig. 3.** TSGA flowchart

**Crossover Map Operator.** As shown in Fig. 4(a), the crossover map operator is used for changing the mapping processor of two chromosomes. The crossover map operator chooses two chromosomes $S$ and $T$ from the population and an integer $I$ between 1 and $N$ randomly. TSGA keeps the processor assignment which is located on the left of $I$. For the processors on the right of $I$, we exchange the processors of $S$ and $T$ which are assigned to execute the same task. The processor assignments of tasks $t_5$, $t_6$, and $t_9$ are exchanging directly, since those tasks are occupied in the same place in both chromosomes. On the other hand, tasks $t_7$ and $t_8$ are located at different places in these two chromosomes, so they are scheduled to the processor in which they are assigned to another chromosome, and TSGA exchanges the processor assignments. The chromosome $S''$ and $T''$ are generated by the crossover map operator in SGA.

**Crossover Order Operator.** The crossover order operator is practiced to the second part of the chromosome. In Fig. 4(b), after choosing two chromosomes $S$ and $T$, TSGA chooses a crossover point $I$ between 1 and $N$. Then, Step 1 copies the left portion of $I$ in $S$ and $T$ to the new chromosome $S'$ and $T'$, respectively. In order to complete the right